AI ASSISTED CODING LAB

ASSIGNMENT-10.2

Name: Guangsinlung Phaomei

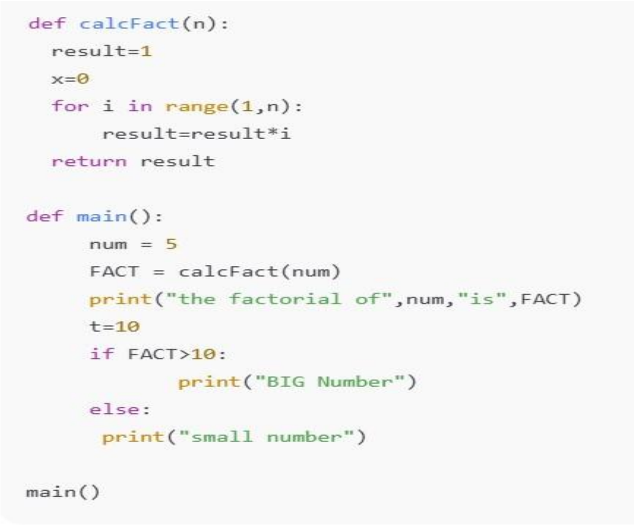
Enroll no: 2503A51L20

Batch: 19

TASK DESCRIPTION 1:

- Write python program as shown below.

- Use an AI assistant to review and suggest corrections.

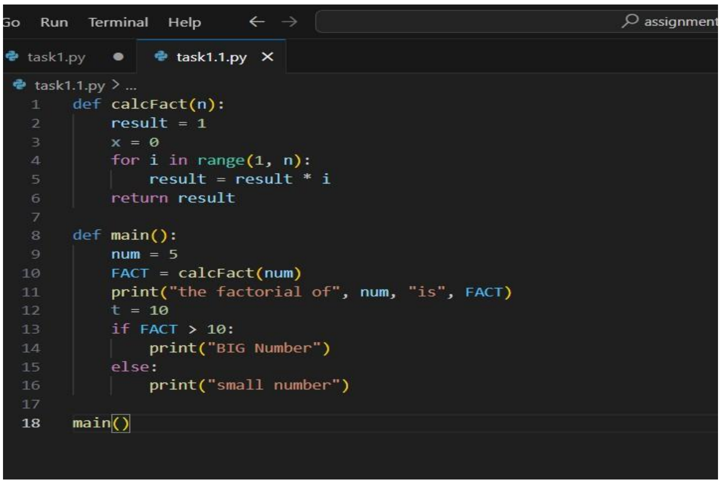


```
def calcFact(n):
  result=1
  x=0
  for i in range(1,n):
      result=result*i
  return result

def main():
     num = 5
     FACT = calcFact(num)
     print("the factorial of",num,"is",FACT)
     t=10
     if FACT>10:
            print("BIG Number")
     else:
      print("small number")

main()
```

PROMPT 1:

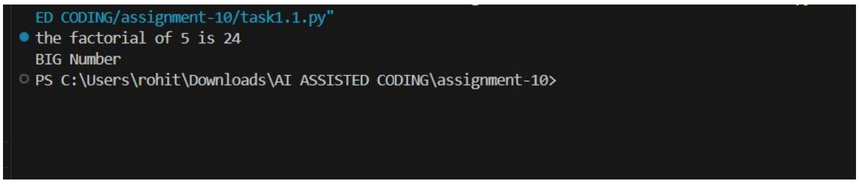
Generate a Python program that calculates the factorial of a number using a function. Then, use a conditional statement to print whether the result is a "BIG Number" or a "small number" based on a threshold.

CODE GENERATED:



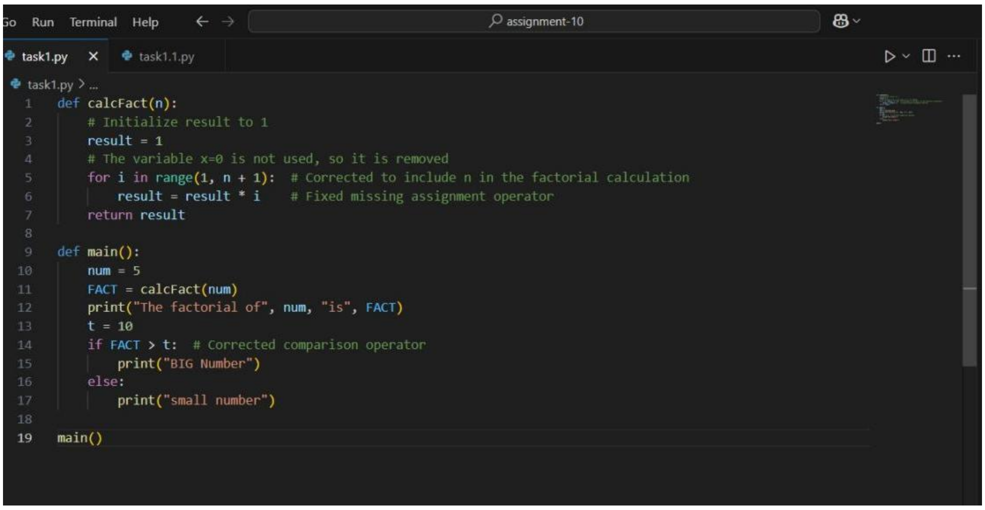
```python
def calcFact(n):
    result = 1
    x = 0
    for i in range(1, n):
        result = result * i
    return result

def main():
    num = 5
    FACT = calcFact(num)
    print("the factorial of", num, "is", FACT)
    t = 10
    if FACT > 10:
        print("BIG Number")
    else:
        print("small number")

main()
```

OUTPUT:



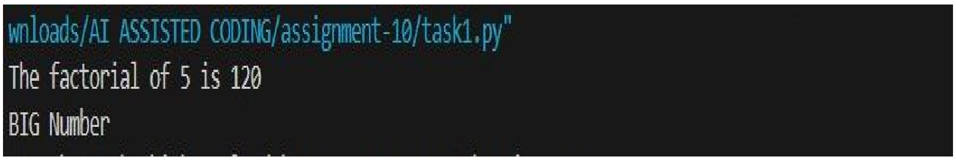
```
ED CODING/assignment-10/task1.1.py"
● the factorial of 5 is 24
  BIG Number
○ PS C:\Users\rohit\Downloads\AI ASSISTED CODING\assignment-10>
```

CORRECT CODE GENERATED:



```python
def calcFact(n):
    # Initialize result to 1
    result = 1
    # The variable x=0 is not used, so it is removed
    for i in range(1, n + 1):  # Corrected to include n in the factorial calculation
        result = result * i    # Fixed missing assignment operator
    return result

def main():
    num = 5
    FACT = calcFact(num)
    print("The factorial of", num, "is", FACT)
    t = 10
    if FACT > t:  # Corrected comparison operator
        print("BIG Number")
    else:
        print("small number")

main()
```

OUTPUT:



```
wnloads/AI ASSISTED CODING/assignment-10/task1.py"
The factorial of 5 is 120
BIG Number
```

OBSERVATION:

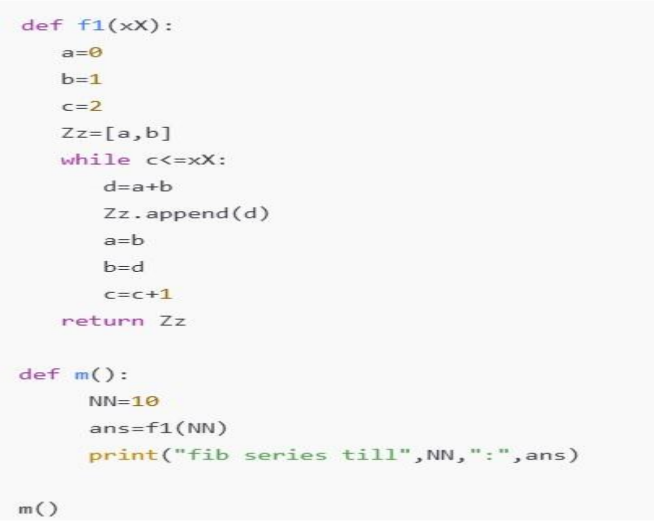
I created a Python program using functions, loops, and conditionals to calculate factorials. With AI help, I improved logic and readability, showing a clear understanding of Python basics

TASK DESCRIPTION 2:

Write the Python code for Fibonacci as shown below and execute.

Ask AI to improve variable names, add comments, and apply PEP8 formatting (cleaned up).

Students evaluate which suggestions improve readability most. one.

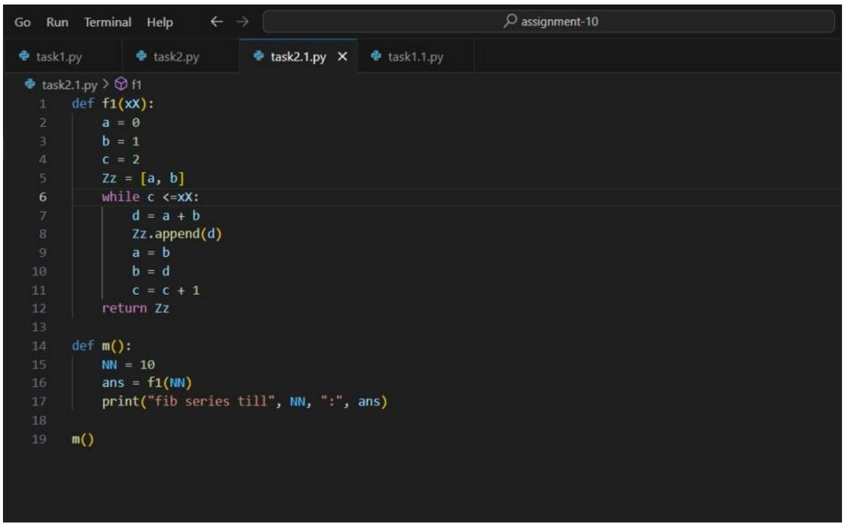


```
def f1(xX):
   a=0
   b=1
   c=2
   Zz=[a,b]
   while c<=xX:
      d=a+b
      Zz.append(d)
      a=b
      b=d
      c=c+1
   return Zz

def m():
     NN=10
     ans=f1(NN)
     print("fib series till",NN,":",ans)

m()
```

PROMPT 1:

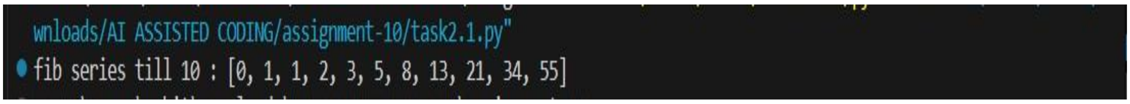
Write a Python program using a function to generate the Fibonacci series up to a given number of terms, then review the code with an AI assistant.

CODE GENERATED:



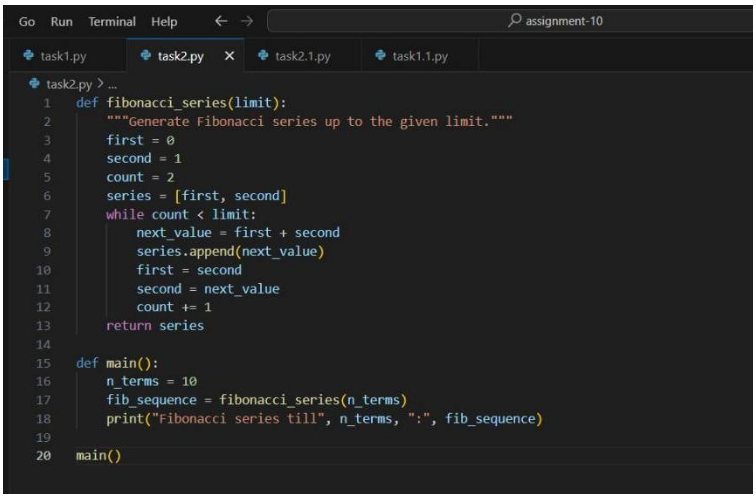
```
def f1(xX):
    a = 0
    b = 1
    c = 2
    Zz = [a, b]
    while c <=xX:
        d = a + b
        Zz.append(d)
        a = b
        b = d
        c = c + 1
    return Zz

def m():
    NN = 10
    ans = f1(NN)
    print("fib series till", NN, ":", ans)

m()
```

OUTPUT:



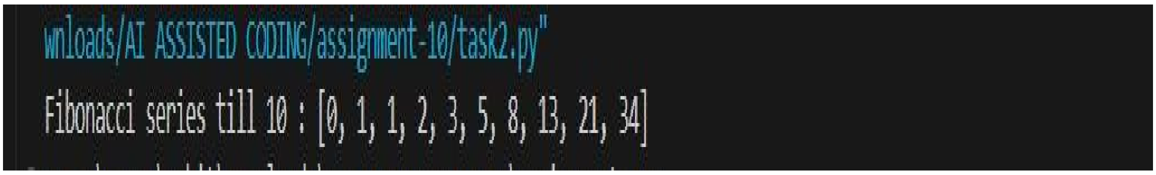
```
wnloads/AI ASSISTED CODING/assignment-10/task2.1.py"
fib series till 10 : [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

COREECT CODE GENERATED:



```python
def fibonacci_series(limit):
    """Generate Fibonacci series up to the given limit."""
    first = 0
    second = 1
    count = 2
    series = [first, second]
    while count < limit:
        next_value = first + second
        series.append(next_value)
        first = second
        second = next_value
        count += 1
    return series

def main():
    n_terms = 10
    fib_sequence = fibonacci_series(n_terms)
    print("Fibonacci series till", n_terms, ":", fib_sequence)

main()
```

OUTPUT:



```
wnloads/AI ASSISTED CODING/assignment-10/task2.py"
Fibonacci series till 10 : [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

OBSERVATION:

I created a Python program to generate the Fibonacci series using a function and loop. With AI feedback, I improved variable naming, formatting, and code clarity, enhancing readability and showing strong understanding and improvement through feedback

TASK DESCRIPTION 3:

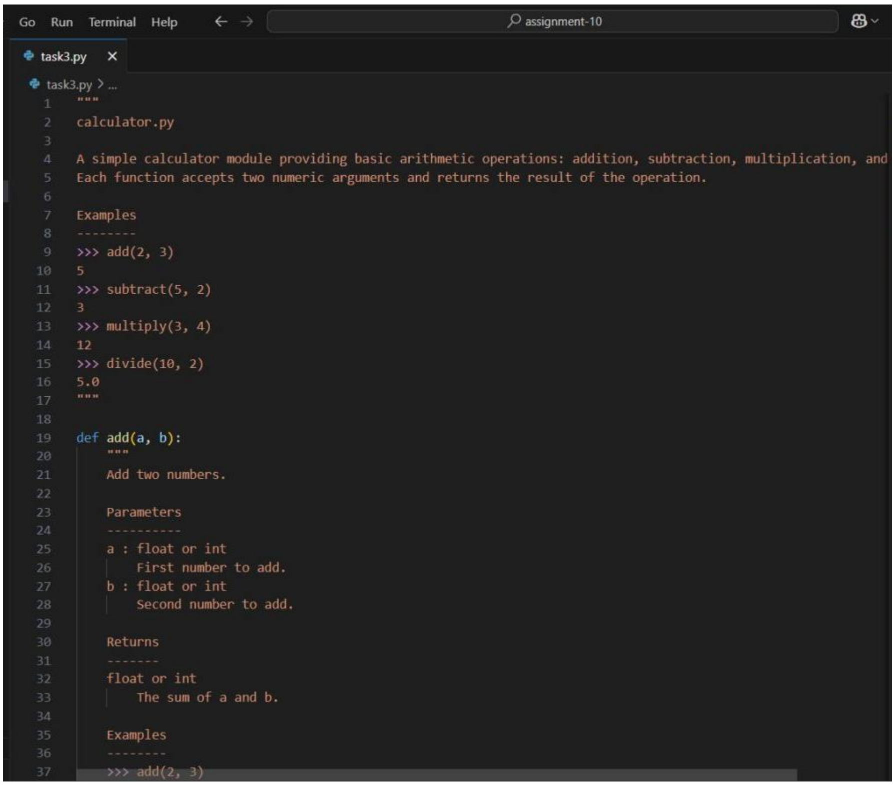
- Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).
- Incorporate manual **docstring** in code with NumPy Style
- Use AI assistance to generate a module-level docstring + individual function docstrings.

- Compare the AI-generated docstring with your manually written one.

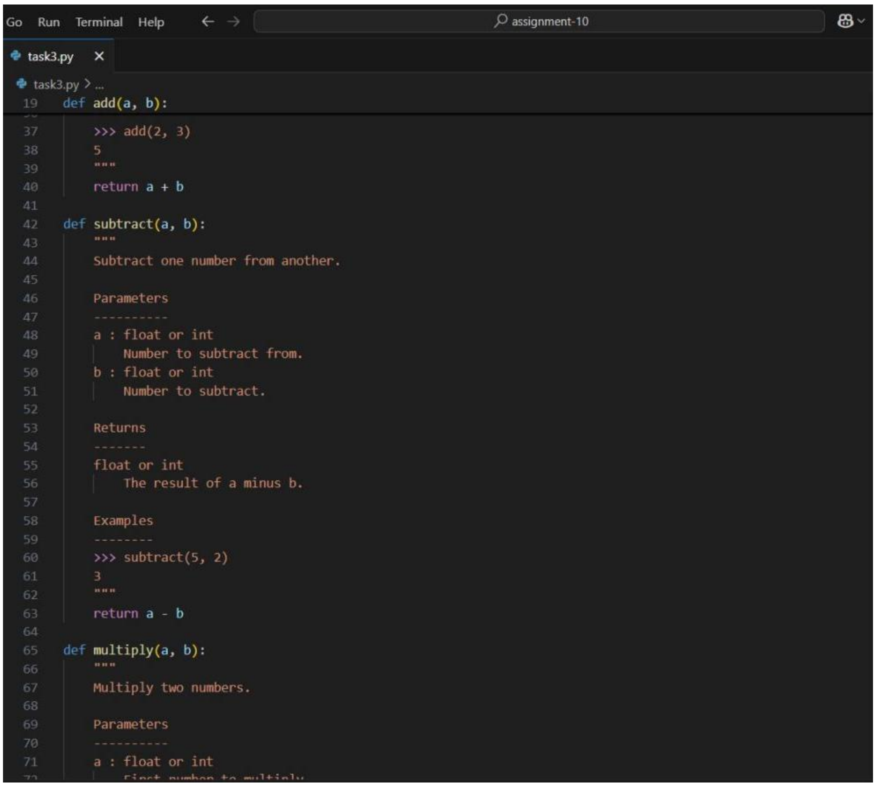
PROMPT 1:

Generate a Python program that does basic math (like adding, subtracting, multiplying, dividing) and practice writing clear documentation for your code.

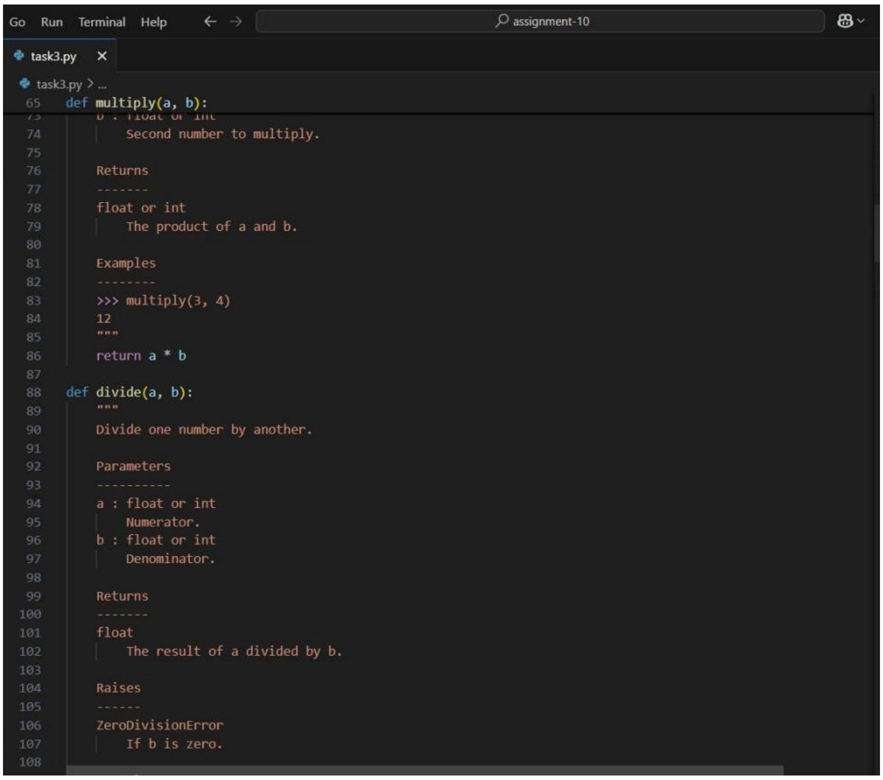
CODE GENERATED:



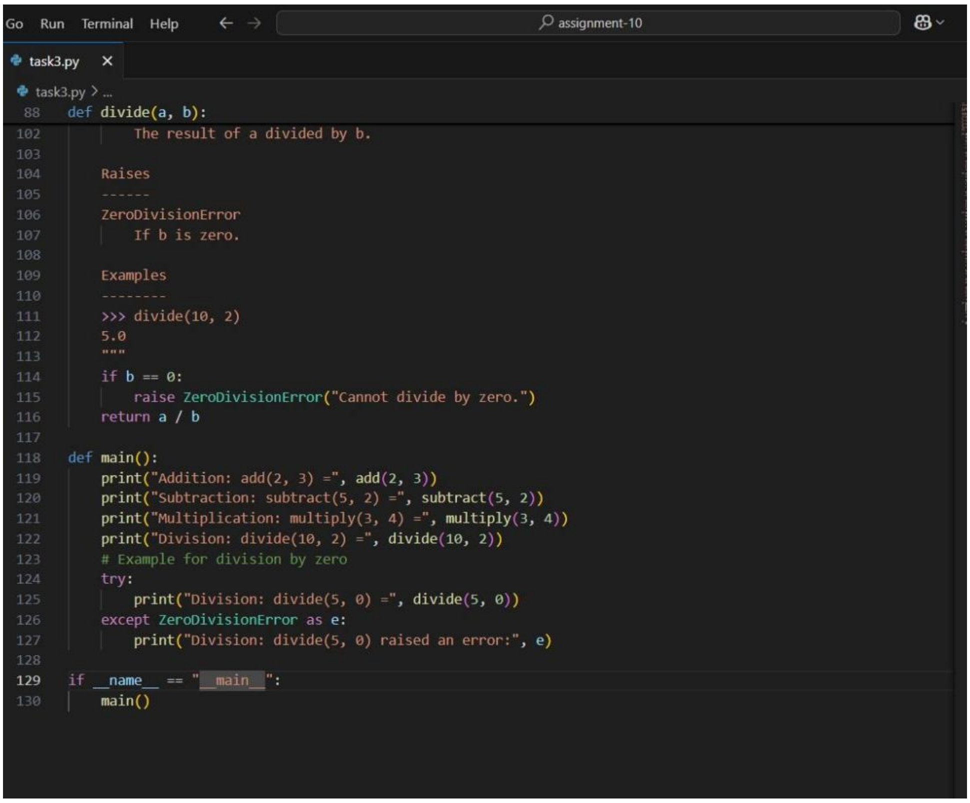
```
"""
calculator.py

A simple calculator module providing basic arithmetic operations: addition, subtraction, multiplication, and
Each function accepts two numeric arguments and returns the result of the operation.

Examples
--------
>>> add(2, 3)
5
>>> subtract(5, 2)
3
>>> multiply(3, 4)
12
>>> divide(10, 2)
5.0
"""


def add(a, b):
    """
    Add two numbers.

    Parameters
    ----------
    a : float or int
        First number to add.
    b : float or int
        Second number to add.

    Returns
    -------
    float or int
        The sum of a and b.

    Examples
    --------
    >>> add(2, 3)
```



```python
def add(a, b):
    >>> add(2, 3)
    5
    """
    return a + b

def subtract(a, b):
    """
    Subtract one number from another.

    Parameters
    ----------
    a : float or int
        Number to subtract from.
    b : float or int
        Number to subtract.

    Returns
    -------
    float or int
        The result of a minus b.

    Examples
    --------
    >>> subtract(5, 2)
    3
    """
    return a - b

def multiply(a, b):
    """
    Multiply two numbers.

    Parameters
    ----------
    a : float or int
        First number to multiply.
```

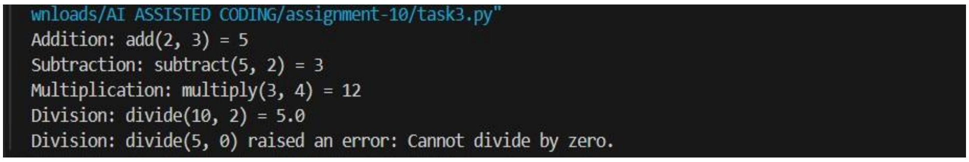


```python
def multiply(a, b):
    b : float or int
        Second number to multiply.

    Returns
    -------
    float or int
        The product of a and b.

    Examples
    --------
    >>> multiply(3, 4)
    12
    """
    return a * b

def divide(a, b):
    """
    Divide one number by another.

    Parameters
    ----------
    a : float or int
        Numerator.
    b : float or int
        Denominator.

    Returns
    -------
    float
        The result of a divided by b.

    Raises
    ------
    ZeroDivisionError
        If b is zero.
```



```
def divide(a, b):
        The result of a divided by b.

    Raises
    ------
    ZeroDivisionError
        If b is zero.

    Examples
    --------
    >>> divide(10, 2)
    5.0
    """
    if b == 0:
        raise ZeroDivisionError("Cannot divide by zero.")
    return a / b

def main():
    print("Addition: add(2, 3) =", add(2, 3))
    print("Subtraction: subtract(5, 2) =", subtract(5, 2))
    print("Multiplication: multiply(3, 4) =", multiply(3, 4))
    print("Division: divide(10, 2) =", divide(10, 2))
    # Example for division by zero
    try:
        print("Division: divide(5, 0) =", divide(5, 0))
    except ZeroDivisionError as e:
        print("Division: divide(5, 0) raised an error:", e)

if __name__ == "__main__":
    main()
```

OUTPUT:



```
wnloads/AI ASSISTED CODING/assignment-10/task3.py"
Addition: add(2, 3) = 5
Subtraction: subtract(5, 2) = 3
Multiplication: multiply(3, 4) = 12
Division: divide(10, 2) = 5.0
Division: divide(5, 0) raised an error: Cannot divide by zero.
```

OBSERVATION:

This assignment combines Python function design with good documentation practices. It reinforces core concepts like functions and parameters while promoting clear, professional NumPy-style docstrings. Comparing manual and AI-generated docstrings improved understanding of clarity and formatting.